TGM HIT diploma thesis template

v0.0.1

https://github.com/TGM-HIT/typst-diploma-thesis

Clemens Koza

Abstract

A diploma thesis template for students of the HIT department at TGM Wien.

Contents

I Introduction 2
II Module reference 2
II.a `tgm-hit-thesis` 2
II.b `tgm-hit-thesis.utils` 4
II.c `tgm-hit-thesis.glossary` 5
II.d `tgm-hit-thesis.l10n` 6
II.e `tgm-hit-thesis.assets` 6

I Introduction

This template is aimed at students of the information technology department at the TGM technical secondary school in Vienna. It can be used both in the Typst app and using the CLI:

Using the Typst web app, you can create a project by e.g. using this link: https://typst.app/?template=tgm-hit-thesis&version=latest.

To work locally, use the following command:

```bash
typst init @preview/tgm-hit-thesis
```

If you are getting started writing your thesis, you will likely be better off looking into the thesis document created by the template: it contains instruction and examples on the most important features of this template. If you have not yet initialized the template, a rendered version is linked in the README. If you are new to Typst, also check out the Typst documentation: https://typst.app/docs/.

The rest of this manual documents the individual functions offered by this package. This is useful if you want to know what customization options are available, or you're not sure what parts of the template package do.

As a school-specific template, this package does not offer enough configurability to be used at different institutions. However, if you like this template, feel free to adapt the code (MIT-licensed) to your needs, or open a Github issue if you think the template could be adapted to work for your requirements.

II Module reference

II.a `tgm-hit-thesis`

The template's main module. All functions that need to be called are directly exported from this module.

- thesis()
- declaration()
- abstract()
- main-matter()

```
thesis(
    title: content string,
    subtitle: content string,
    authors: array,
    supervisor-label: content string auto,
    supervisor: content string,
    date: datetime,
    year: content string,
    division: content string,
    logo: content,
    bibliography: content,
    language: string,
    paper: string,
    strict-chapter-end: bool
  ) -> function
```

The main template function. Your document will generally start with `#show: thesis(...)`, which it already does after initializing the template. Although all parameters are named, most of them are

really mandatory. Parameters that are not given may result in missing content in places where it is not actually optional.

Parameters:

`title` (`content` or `string` = `none`) – The title of the thesis, displayed on the title page and used for PDF metadata.

`subtitle` (`content` or `string` = `none`) – A descriptive one-liner that gives the reader an immediate idea about the thesis' topic.

`authors` (`array` = `()`) – The thesis authors. Each array entry is a `dict` of the form `(name: ..., class: ..., subtitle: ...)` stating their name, class, and the title of their part of the whole thesis project. The names must be regular strings, for the PDF metadata.

`supervisor-label` (`content` or `string` or `auto` = `auto`) – The term with which to label the supervisor name; if not given or `auto`, this defaults to a language-dependent text. In German, this text is gender-specific and can be overridden with this parameter.

`supervisor` (`content` or `string` = `none`) – The name of the thesis' supervisor.

`date` (`datetime` = `none`) – The date of submission of the thesis.

`year` (`content` or `string` = `none`) – The school year in which the thesis was produced.

`division` (`content` or `string` = `none`) – The division inside the HIT department (i.e. usually "Medientechnik" or "Systemtechnik").

`logo` (`content` = `none`) – The image (`image()`) to use as the document's logo on the title page.

`bibliography` (`content` = `none`) – The bibliography (`bibliography()`) to use for the thesis.

`language` (`string` = `"de"`) – The language in which the thesis is written. `"de"` and `"en"` are supported. The choice of language influences certain texts on the title page and in headings, as well as the date format used on the title page.

`paper` (`string` = `"a4"`) – Changes the paper format of the thesis. Use this option with care, as it will shift various contents around.

`strict-chapter-end` (`bool` = `false`) – This can be activated to ensure proper use of the `chapter-end()` function. It is disabled by default because it slows down compilation. See `enforce-chapter-end-placement()`.

declaration(signature-height: `length`, body: `content`) -> `content`

The statutory declaration that the thesis was written without improper help. The text is not part of the template so that it can be adapted according to one's needs. Example texts are given in the template. Heading and signature lines for each author are inserted automatically.

Parameters:

`signature-height` (`length` = `1.1cm`) – The height of the signature line. The default should be able to fit up to seven authors on one page; for larger teams, the height can be decreased.

`body` (`content`) – The actual declaration.

abstract(lang: `string`, body: `content`)

An abstract section. This should appear twice in the thesis regardless of language; first for the German *Kurzfassung*, then for the English abstract.

Parameters:

`lang` (`string` = `auto`) – The language of this abstract. Although it defaults to `auto`, in which case the document's language is used, it's preferable to always set the language explicitly.

`body` (`content`) – The abstract text.

main-matter() -> `function`

Starts the main matter of the thesis. This should be called as a show rule (`#show: main-matter()`) after the abstracts and will insert the table of contents. All subsequent top level headings will be treated as chapters and thus be numbered and outlined.

II.b `tgm-hit-thesis.utils`

Utilities, mostly internal. The `chapter-end()` function is re-exported from the main module.

- is-chapter-page()
- chapter-end()
- is-empty-page()
- enforce-chapter-end-placement()
- is-first-section()
- repeat()

is-chapter-page() -> `bool`

Internal function. Returns whether the current page is one where a chapter begins. This is used for styling headers and footers.

This function is contextual.

chapter-end() -> `content`

Inserts an invisible marker that marks the end of a chapter. This is used for determining whether a page is empty and thus whether it should have header and footer.

is-empty-page() -> `bool`

Internal function. Returns whether the current page is empty. This is determined by checking whether the current page is between the previous chapter's end and the next chapter's beginning. This is used for styling headers and footers.

This function is contextual.

enforce-chapter-end-placement() -> `content`

Internal function. Checks whether all chapters and chapter ends are placed properly. The document should contain an alternating sequence of chapters and chapter ends; if a chapter doesn't have an end or vice-versa, this can lead to wrongly displayed/hidden headers and footers.

The result of this function is invisible if it succeeds.

is-first-section() -> `bool`

Internal function. This is intended to be called in a section show rule. It returns whether that section is the first in the current chapter

This function is contextual.

repeat(gap: `length` `none`, justify: `bool`, body: `content`) -> `content`

Repeat the given content to fill the full space. In contrast to the built-in `repeat()` function, this can produce exact and aligned gaps.

Parameters:

`gap` (`length` or `none` = `none`) – The gap between repeated items.

`justify` (`bool` = `false`) – Whether to increase the gap to justify the items.

`body` (`content`) – the content to repeat

II.c `tgm-hit-thesis.glossary`

Wrappers for Glossarium functionality. The `glossary-entry()` function and Glossarium's `gls()` and `glspl()` are re-exported from the main module.

- glossary-entry()
- print-glossary()

```
glossary-entry(
    key: string,
    short: string,
    long: string content,
    desc: string content,
    plural: string content,
    longplural: string content,
    group: string
  ) -> content
```

Stores a glossary entry for this thesis. One call to this function is equivalent to one array entry in Glossarium's `print-glossary()`'s main parameter.

Parameters:

`key` (`string`) – The key with which the glossary entry can be referenced; must be unique.

`short` (`string` = `none`) – Mandatory; the short form of the entry shown after the term has been first defined.

`long` (`string` or `content` = `none`) – The long form of the term, displayed in the glossary and on the first citation of the term.

`desc` (`string` or `content` = `none`) – The description of the term.

`plural` (`string` or `content` = `none`) – The pluralized short form of the term.

`longplural` (`string` or `content` = `none`) – The pluralized long form of the term.

`group` (`string` = `none`) – The group the term belongs to. The terms are displayed by groups in the glossary.

print-glossary(title: `content`, ..args: `arguments`)

Displays a glossary of the entries added via `glossary-entry()`.

Parameters:

`title` (`content` = `none`) – A (level 1) heading that titles this glossary. If the glossary is empty, the title is not shown.

`..args` (`arguments`) – Any extra parameters to the glossarium function of the same name.

II.d `tgm-hit-thesis.l10n`

Contains contextual constants that display localized strings. This is a thin wrapper around linguify that improves autocomplete and avoids typos in Typst code. Have a look at the source code to see what definitions are available.

- set-database()

set-database() -> `content`

Internal function. Initializes Linguify with the template's translation file.

II.e `tgm-hit-thesis.assets`

Contains images used in the template.

Variables:

- tgm-logo
- htl-logo

tgm-logo

The TGM logo. This is a partially applied function and thus can accept most of the parameters that `image()` can.

htl-logo

The HTL logo. This is a partially applied function and thus can accept most of the parameters that `image()` can.